

```

EVENT: Start with the library "bags".

;; Tue Sep 26 10:20:45 1989, from ~wilding/numerical/newnat.events

;; NATURALS Theory

;; Created by Bill Bevier 1988 (see CLI internal note 057)

;; Modifications by Bill Bevier and Matt Wilding (9/89) including
;; adding some new metalemmas for times, reorganizing the theories,
;; removing some extraneous lemmas, and removing dependence upon
;; other theories (by adding the pertinent lemmas).

;; This script requires the bags theory

;; This script sets up a theory for the NATURALS with the following subtheories
;; ADDITION
;; MULTIPLICATION
;; REMAINDER
;; QUOTIENT
;; EXPONENTIATION
;; LOGS
;; GCDS

;; The theories of EXPONENTIATION, LOGS, and GCDS still need a lot of work

; -----
; ARITHMETIC
; -----

; ----- PLUS & DIFFERENCE -----

; ----- EQUAL -----

THEOREM: equal-plus-0
 $((a + b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$ 

THEOREM: plus-cancellation
 $((a + b) = (a + c)) = (\text{fix}(b) = \text{fix}(c))$ 

EVENT: Disable plus-cancellation.

THEOREM: equal-difference-0

```

$$(((x - y) = 0) = (y \not< x)) \wedge ((0 = (x - y)) = (y \not< x))$$

THEOREM: difference-cancellation

$$\begin{aligned} & ((x - y) = (z - y)) \\ = & \text{ if } x < y \text{ then } y \not< z \\ & \text{ elseif } z < y \text{ then } y \not< x \\ & \text{ else } \text{fix}(x) = \text{fix}(z) \text{ endif} \end{aligned}$$

EVENT: Disable difference-cancellation.

; ----- PLUS -----

THEOREM: commutativity-of-plus

$$(x + y) = (y + x)$$

THEOREM: commutativity2-of-plus

$$(x + (y + z)) = (y + (x + z))$$

THEOREM: plus-zero-arg2

$$(y \simeq 0) \rightarrow ((x + y) = \text{fix}(x))$$

THEOREM: plus-add1-arg1

$$((1 + a) + b) = (1 + (a + b))$$

THEOREM: plus-add1-arg2

$$\begin{aligned} & (x + (1 + y)) \\ = & \text{ if } y \in \mathbf{N} \text{ then } 1 + (x + y) \\ & \text{ else } 1 + x \text{ endif} \end{aligned}$$

THEOREM: associativity-of-plus

$$((x + y) + z) = (x + (y + z))$$

THEOREM: plus-difference-arg1

$$\begin{aligned} & ((a - b) + c) \\ = & \text{ if } b < a \text{ then } (a + c) - b \\ & \text{ else } 0 + c \text{ endif} \end{aligned}$$

THEOREM: plus-difference-arg2

$$\begin{aligned} & (a + (b - c)) \\ = & \text{ if } c < b \text{ then } (a + b) - c \\ & \text{ else } a + 0 \text{ endif} \end{aligned}$$

; ----- DIFFERENCE-PLUS cancellation rules -----

;

; Here are the basic canonicalization rules for differences of sums. These
; are subsumed by the meta lemmas and are therefore globally disabled.

; They are here merely to prove the meta lemmas.

THEOREM: difference-plus-cancellation-proof

$$((x + y) - x) = \text{fix}(y)$$

THEOREM: difference-plus-cancellation

$$(((x + y) - x) = \text{fix}(y)) \wedge (((y + x) - x) = \text{fix}(y))$$

EVENT: Disable difference-plus-cancellation.

THEOREM: difference-plus-plus-cancellation-proof

$$((x + y) - (x + z)) = (y - z)$$

THEOREM: difference-plus-plus-cancellation

$$(((x + y) - (x + z)) = (y - z))$$

$$\wedge (((y + x) - (x + z)) = (y - z))$$

$$\wedge (((x + y) - (z + x)) = (y - z))$$

$$\wedge (((y + x) - (z + x)) = (y - z))$$

EVENT: Disable difference-plus-plus-cancellation.

THEOREM: difference-plus-plus-cancellation-hack

$$((w + x + a) - (y + z + a)) = ((w + x) - (y + z))$$

EVENT: Disable difference-plus-plus-cancellation-hack.

; Here are a few more facts about difference needed to prove the meta lemmas.
; These are disabled here. We re-prove them after the proof of the meta
; lemmas so that they will fire before the meta lemmas in subsequent proofs.

THEOREM: diff-sub1-arg2

$$(a - (b - 1))$$

$$= \text{if } b \simeq 0 \text{ then } \text{fix}(a)$$

$$\text{elseif } a < b \text{ then } 0$$

$$\text{else } 1 + (a - b) \text{ endif}$$

EVENT: Disable diff-sub1-arg2.

THEOREM: diff-diff-arg1

$$((x - y) - z) = (x - (y + z))$$

THEOREM: diff-diff-arg2

$$(a - (b - c))$$

$$= \text{if } b < c \text{ then } \text{fix}(a)$$

$$\text{else } (a + c) - b \text{ endif}$$

```
; diff-diff-diff should be removed, but since the hack lemmas for
; correctness-of-cancel-difference-plus are designed for it, we'll
; keep it around.
```

THEOREM: diff-diff-diff

$$\begin{aligned} & ((b \leq a) \wedge (d \leq c)) \\ \rightarrow & (((a - b) - (c - d)) = ((a + d) - (b + c))) \end{aligned}$$

EVENT: Disable diff-diff-diff.

THEOREM: difference-lessp-arg1

$$(a < b) \rightarrow ((a - b) = 0)$$

EVENT: Disable difference-lessp-arg1.

```
; -----
; Meta Lemmas to Cancel PLUS and DIFFERENCE expressions
; -----
```

```
; ----- PLUS-TREE and PLUS-FRINGER -----
```

DEFINITION:

```
plus-fringe (x)
= if listp (x) and (car (x) = 'plus)
  then append (plus-fringe (cadr (x)), plus-fringe (caddr (x)))
  else cons (x, nil) endif
```

DEFINITION:

```
plus-tree (l)
= if l  $\simeq$  nil then '0
  elseif cdr (l)  $\simeq$  nil then list ('fix, car (l))
  elseif caddr (l)  $\simeq$  nil then list ('plus, car (l), cadr (l))
  else list ('plus, car (l), plus-tree (cdr (l))) endif
```

THEOREM: numberp-eval\$-plus

$$(\text{listp}(x) \wedge (\text{car}(x) = \text{'plus})) \rightarrow (\text{eval}\$(t, x, a) \in \mathbf{N})$$

EVENT: Disable numberp-eval\$-plus.

THEOREM: numberp-eval\$-plus-tree

$$\text{eval}\$(t, \text{plus-tree}(l), a) \in \mathbf{N}$$

EVENT: Disable numberp-eval\$-plus-tree.

THEOREM: member-implies-plus-tree-greatereqp
 $(x \in y) \rightarrow (\text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a) \not\leq \text{eval}\$(\mathbf{t}, x, a))$

EVENT: Disable member-implies-plus-tree-greatereqp.

THEOREM: plus-tree-delete
 $\text{eval}\$(\mathbf{t}, \text{plus-tree}(\text{delete}(x, y)), a)$
 $=$ **if** $x \in y$ **then** $\text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a) - \text{eval}\(\mathbf{t}, x, a)
else $\text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a)$ **endif**

EVENT: Disable plus-tree-delete.

THEOREM: subbagp-implies-plus-tree-greatereqp
 $\text{subbagp}(x, y) \rightarrow (\text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a) \not\leq \text{eval}\$(\mathbf{t}, \text{plus-tree}(x), a))$

EVENT: Disable subbagp-implies-plus-tree-greatereqp.

THEOREM: plus-tree-bagdiff
 $\text{subbagp}(x, y)$
 $\rightarrow (\text{eval}\$(\mathbf{t}, \text{plus-tree}(\text{bagdiff}(y, x)), a)$
 $= (\text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a) - \text{eval}\$(\mathbf{t}, \text{plus-tree}(x), a)))$

EVENT: Disable plus-tree-bagdiff.

THEOREM: numberp-eval\$-bridge
 $(\text{eval}\$(\mathbf{t}, z, a) = \text{eval}\$(\mathbf{t}, \text{plus-tree}(x), a)) \rightarrow (\text{eval}\$(\mathbf{t}, z, a) \in \mathbf{N})$

EVENT: Disable numberp-eval\$-bridge.

THEOREM: bridge-to-subbagp-implies-plus-tree-greatereqp
 $(\text{subbagp}(y, \text{plus-fringe}(z))$
 $\wedge (\text{eval}\$(\mathbf{t}, z, a) = \text{eval}\$(\mathbf{t}, \text{plus-tree}(\text{plus-fringe}(z)), a)))$
 $\rightarrow ((\text{eval}\$(\mathbf{t}, z, a) < \text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a)) = \mathbf{f})$

EVENT: Disable bridge-to-subbagp-implies-plus-tree-greatereqp.

THEOREM: eval\$-plus-tree-append
 $\text{eval}\$(\mathbf{t}, \text{plus-tree}(\text{append}(x, y)), a)$
 $= (\text{eval}\$(\mathbf{t}, \text{plus-tree}(x), a) + \text{eval}\$(\mathbf{t}, \text{plus-tree}(y), a))$

EVENT: Disable eval\$-plus-tree-append.

THEOREM: plus-tree-plus-fringe
 $\text{eval}\$(\mathbf{t}, \text{plus-tree}(\text{plus-fringe}(x)), a) = \text{fix}(\text{eval}\$(\mathbf{t}, x, a))$

EVENT: Disable plus-tree-plus-fringe.

THEOREM: member-implies-numberp
 $((c \in \text{plus-fringe}(x)) \wedge (\text{eval}\$(\mathbf{t}, c, a) \in \mathbf{N})) \rightarrow (\text{eval}\$(\mathbf{t}, x, a) \in \mathbf{N}))$

EVENT: Disable member-implies-numberp.

THEOREM: cadr-eval\$list
 $(\text{car}(\text{eval}\$('list, x, a)) = \text{eval}\$(\mathbf{t}, \text{car}(x), a))$
 $\wedge (\text{cdr}(\text{eval}\$('list, x, a))$
 $\quad = \text{if listp}(x) \text{ then eval}\$('list, \text{cdr}(x), a)$
 $\quad \text{else } 0 \text{ endif})$

EVENT: Disable cadr-eval\$list.

THEOREM: eval\$-quote
 $\text{eval}\$(\mathbf{t}, \text{cons}('quote, args), a) = \text{car}(args)$

EVENT: Disable eval\$-quote.

THEOREM: listp-eval\$
 $\text{listp}(\text{eval}\$('list, x, a)) = \text{listp}(x)$

EVENT: Disable listp-eval\$.

; ----- CANCEL PLUS -----

; CANCEL-EQUAL-PLUS cancels identical terms in a term which is the equality
; of two sums. For example,
;
; (EQUAL (PLUS A B C) (PLUS B D E)) => (EQUAL (PLUS A C) (PLUS D E))
;
;

DEFINITION:
cancel-equal-plus(x)
 $= \text{if listp}(x) \wedge (\text{car}(x) = 'equal)$

```

then if listp(cadr(x))
  ^ (caadr(x) = 'plus)
  ^ listp(caddr(x))
  ^ (caaddr(x) = 'plus)
then list('equal,
  plus-tree(bagdiff(plus-fringe(cadr(x)),
                  bagint(plus-fringe(cadr(x)),
                        plus-fringe(caddr(x))))),
  plus-tree(bagdiff(plus-fringe(caddr(x)),
                  bagint(plus-fringe(cadr(x)),
                        plus-fringe(caddr(x))))))

elseif listp(cadr(x))
  ^ (caadr(x) = 'plus)
  ^ (caddr(x) ∈ plus-fringe(cadr(x)))
then list('if,
  list('numberp, caddr(x)),
  list('equal,
  plus-tree(delete(caddr(x), plus-fringe(cadr(x))),
  '0),
  list('quote, f))
elseif listp(caddr(x))
  ^ (caaddr(x) = 'plus)
  ^ (cadr(x) ∈ plus-fringe(caddr(x)))
then list('if,
  list('numberp, cadr(x)),
  list('equal,
  '0,
  plus-tree(delete(cadr(x), plus-fringe(caddr(x))))),
  list('quote, f))
else x endif
else x endif

```

THEOREM: correctness-of-cancel-equal-plus
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-equal-plus}(x), a)$

```

; ----- CANCEL-DIFFERENCE-PLUS -----
;
; CANCEL-DIFFERENCE-PLUS cancels identical terms in a term which is the
; difference of two sums. For example,
;
; (DIFFERENCE (PLUS A B C) (PLUS B D E)) => (DIFFERENCE (PLUS A C) (PLUS D E))
;
; Using rewrite rules, we canonicalize terms involving PLUS and DIFFERENCE
; to be the DIFFERENCE of two sums. Then CANCEL-DIFFERENCE-PLUS cancels out

```

; like terms.

DEFINITION:

```

cancel-difference-plus(x)
=  if listp(x) ∧ (car(x) = 'difference)
    then if listp(cadr(x))
          ∧ (caadr(x) = 'plus)
          ∧ listp(caddr(x))
          ∧ (caaddr(x) = 'plus)
        then list('difference,
                  plus-tree(bagdiff(plus-fringe(cadr(x)),
                                   bagint(plus-fringe(cadr(x)),
                                         plus-fringe(caddr(x))))),
                  plus-tree(bagdiff(plus-fringe(caddr(x)),
                                   bagint(plus-fringe(cadr(x)),
                                         plus-fringe(caddr(x))))))
        elseif listp(cadr(x))
          ∧ (caadr(x) = 'plus)
          ∧ (caddr(x) ∈ plus-fringe(cadr(x)))
        then plus-tree(delete(caddr(x), plus-fringe(cadr(x))))
        elseif listp(caddr(x))
          ∧ (caaddr(x) = 'plus)
          ∧ (cadr(x) ∈ plus-fringe(caddr(x))) then '0
        else x endif
    else x endif

```

THEOREM: correctness-of-cancel-difference-plus
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-difference-plus}(x), a)$

; ----- DIFFERENCE -----

; Here are the rules for difference terms which we want to try before
the meta lemmas. They help canonicalize terms to differences of sums.

THEOREM: difference-elim
 $((y \in \mathbf{N}) \wedge (y \not\leq x)) \rightarrow ((x + (y - x)) = y)$

THEOREM: difference-leq-arg1
 $(a \leq b) \rightarrow ((a - b) = 0)$

THEOREM: difference-add1-arg2
 $(a - (1 + b))$
= if $b < a$ then $(a - b) - 1$
 else 0 endif

THEOREM: difference-sub1-arg2

```
(a - (b - 1))
=  if b  $\simeq$  0 then fix(a)
    elseif a < b then 0
    else 1 + (a - b) endif
```

THEOREM: difference-difference-arg1

```
((x - y) - z) = (x - (y + z))
```

THEOREM: difference-difference-arg2

```
(a - (b - c))
=  if b < c then fix(a)
    else (a + c) - b endif
```

THEOREM: difference-x-x

```
(x - x) = 0
```

; ----- LESSP -----

THEOREM: lessp-difference-cancellation

```
((a - c) < (b - c))
=  if c  $\leq$  a then a < b
    else c < b endif
```

EVENT: Disable lessp-difference-cancellation.

```
; CANCEL-LESSP-PLUS cancels LESSP terms whose arguments are sums.
; Examples:
; (LESSP (PLUS A B C) (PLUS A C D)) -> (LESSP (FIX B) (FIX D))
; (LESSP A (PLUS A B)) -> (NOT (ZEROP (FIX B)))
; (LESSP (PLUS A B) A) -> F
```

DEFINITION:

```
cancel-lessp-plus(x)
=  if listp(x)  $\wedge$  (car(x) = 'lessp)
    then if listp(cadr(x))
         $\wedge$  (caadr(x) = 'plus)
         $\wedge$  listp(caddr(x))
         $\wedge$  (caaddr(x) = 'plus)
    then list('lessp,
              plus-tree(bagdiff(plus-fringe(cadr(x)),
                               bagint(plus-fringe(cadr(x)),
                                       plus-fringe(caddr(x)))))),
```

```

      plus-tree (bagdiff (plus-fringe (caddr (x)),
                        bagint (plus-fringe (cadr (x)),
                                plus-fringe (caddr (x))))))
    elseif listp (cadr (x))
      ^ (caadr (x) = 'plus)
      ^ (caddr (x) ∈ plus-fringe (cadr (x)))
    then list ('quote, f)
    elseif listp (caddr (x))
      ^ (caaddr (x) = 'plus)
      ^ (cadr (x) ∈ plus-fringe (caddr (x)))
    then list ('not,
              list ('zerop,
                    plus-tree (delete (cadr (x), plus-fringe (caddr (x))))))
    else x endif
  else x endif

```

THEOREM: correctness-of-cancel-lessp-plus
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-lessp-plus}(x), a)$

```

; Define the available theory of addition. To get the list of events to
; put in the theory, evaluate the following form in NQTHM at this point
; in the script. This form lists all lemmas which are globally enabled,
; and which have non-null lemma type.
;
; (remove-if-not (function (lambda (x)
;   (and (member x (lemmas))
;        (not (assoc x disabled-lemmas))
;        (not (null (nth 2 (get x 'event)))))))
;   chronology)

```

EVENT: Let us define the theory *addition* to consist of the following events: equal-plus-0, equal-difference-0, commutativity-of-plus, commutativity2-of-plus, plus-zero-arg2, plus-add1-arg2, plus-add1-arg1, associativity-of-plus, plus-difference-arg1, plus-difference-arg2, diff-diff-arg1, diff-diff-arg2, correctness-of-cancel-equal-plus, correctness-of-cancel-difference-plus, difference-elim, difference-leq-arg1, difference-add1-arg2, difference-sub1-arg2, difference-difference-arg1, difference-difference-arg2, difference-x-x, correctness-of-cancel-lessp-plus.

```

; ----- TIMES -----

```

THEOREM: equal-times-0
 $((x * y) = 0) = ((x \simeq 0) \vee (y \simeq 0))$

THEOREM: equal-times-1
 $((a * b) = 1) = ((a = 1) \wedge (b = 1))$

```
;(lemma equal-sub1-times-0 (rewrite)
;   (equal (equal (sub1 (times a b)) 0)
;   (or (zerop a)
;   (zerop b)
;   (and (equal a 1) (equal b 1))))))
```

THEOREM: equal-sub1-0
 $((x - 1) = 0) = ((x \simeq 0) \vee (x = 1))$

THEOREM: times-zero
 $(y \simeq 0) \rightarrow ((x * y) = 0)$

THEOREM: times-add1
 $(x * (1 + y))$
 $=$ **if** $y \in \mathbf{N}$ **then** $x + (x * y)$
else $\text{fix}(x)$ **endif**

THEOREM: commutativity-of-times
 $(y * x) = (x * y)$

THEOREM: times-distributes-over-plus-proof
 $(x * (y + z)) = ((x * y) + (x * z))$

THEOREM: times-distributes-over-plus
 $((x * (y + z)) = ((x * y) + (x * z)))$
 $\wedge ((x + y) * z = ((x * z) + (y * z)))$

THEOREM: commutativity2-of-times
 $(x * y * z) = (y * x * z)$

THEOREM: associativity-of-times
 $((x * y) * z) = (x * y * z)$

THEOREM: times-distributes-over-difference-proof
 $((a - b) * c) = ((a * c) - (b * c))$

THEOREM: times-distributes-over-difference
 $((a - b) * c) = ((a * c) - (b * c))$
 $\wedge ((a * (b - c)) = ((a * b) - (a * c)))$

THEOREM: times-quotient-proof
 $((x \neq 0) \wedge ((y \bmod x) = 0)) \rightarrow (((y \div x) * x) = \text{fix}(y))$

THEOREM: times-quotient

$$\begin{aligned} & ((y \neq 0) \wedge ((x \bmod y) = 0)) \\ \rightarrow & (((x \div y) * y) = \text{fix}(x)) \wedge ((y * (x \div y)) = \text{fix}(x)) \end{aligned}$$

THEOREM: times-1-arg1

$$(1 * x) = \text{fix}(x)$$

THEOREM: lessp-times1-proof

$$((a < b) \wedge (c \neq 0)) \rightarrow ((a < (b * c)) = \mathbf{t})$$

THEOREM: lessp-times1

$$\begin{aligned} & ((a < b) \wedge (c \neq 0)) \\ \rightarrow & (((a < (b * c)) = \mathbf{t}) \wedge ((a < (c * b)) = \mathbf{t})) \end{aligned}$$

THEOREM: lessp-times2-proof

$$((a \leq b) \wedge (c \neq 0)) \rightarrow (((b * c) < a) = \mathbf{f})$$

THEOREM: lessp-times2

$$\begin{aligned} & ((a \leq b) \wedge (c \neq 0)) \\ \rightarrow & (((b * c) < a) = \mathbf{f}) \wedge (((c * b) < a) = \mathbf{f}) \end{aligned}$$

THEOREM: lessp-times3-proof1

$$((a \neq 0) \wedge (1 < b)) \rightarrow (a < (a * b))$$

THEOREM: lessp-times3-proof2

$$(a < (a * b)) \rightarrow ((a \neq 0) \wedge (1 < b))$$

THEOREM: lessp-times3

$$\begin{aligned} & ((a < (a * b)) = ((a \neq 0) \wedge (1 < b))) \\ \wedge & ((a < (b * a)) = ((a \neq 0) \wedge (1 < b))) \end{aligned}$$

THEOREM: lessp-times-cancellation-proof

$$((x * z) < (y * z)) = ((z \neq 0) \wedge (x < y))$$

THEOREM: lessp-times-cancellation1

$$\begin{aligned} & (((x * z) < (y * z)) = ((z \neq 0) \wedge (x < y))) \\ \wedge & (((z * x) < (y * z)) = ((z \neq 0) \wedge (x < y))) \\ \wedge & (((x * z) < (z * y)) = ((z \neq 0) \wedge (x < y))) \\ \wedge & (((z * x) < (z * y)) = ((z \neq 0) \wedge (x < y))) \end{aligned}$$

EVENT: Disable lessp-times-cancellation1.

THEOREM: lessp-plus-times-proof

$$(x < a) \rightarrow (((x + (a * b)) < (a * c)) = (b < c))$$

THEOREM: lessp-plus-times1

$((a + (b * c)) < b) = ((a < b) \wedge (c \simeq 0))$
 $\wedge (((a + (c * b)) < b) = ((a < b) \wedge (c \simeq 0)))$
 $\wedge (((c * b) + a) < b) = ((a < b) \wedge (c \simeq 0))$
 $\wedge (((b * c) + a) < b) = ((a < b) \wedge (c \simeq 0))$

THEOREM: lessp-plus-times2

$((a \neq 0) \wedge (x < a))$
 $\rightarrow (((x + (a * b)) < (a * c)) = (b < c))$
 $\wedge (((x + (b * a)) < (a * c)) = (b < c))$
 $\wedge (((x + (a * b)) < (c * a)) = (b < c))$
 $\wedge (((x + (b * a)) < (c * a)) = (b < c))$
 $\wedge (((a * b) + x) < (a * c)) = (b < c)$
 $\wedge (((b * a) + x) < (a * c)) = (b < c)$
 $\wedge (((a * b) + x) < (c * a)) = (b < c)$
 $\wedge (((b * a) + x) < (c * a)) = (b < c)$

THEOREM: lessp-1-times

$(1 < (a * b))$
 $= (\neg((a \simeq 0) \vee (b \simeq 0) \vee ((a = 1) \wedge (b = 1))))$

```
;;; meta lemmas to cancel lessp-times and equal-times expressions
;;; examples
;;; (lessp (times b (times d a)) (times b (times e (times a f)))) ->
;;;                                     (and (and (not (zerop a))
;;;                                     (not (zerop b)))
;;;                                     (lessp (fix d) (times e f)))
;;;
;;; (equal (times b (times c d)) (times b d)) ->
;;;                                     (or (or (zerop b) (zerop d))
;;;                                     (equal (fix c) 1))
;;;
```

DEFINITION:

times-tree(x)
= **if** x \simeq nil **then** ''1
 elseif cdr(x) \simeq nil **then** list('fix, car(x))
 elseif caddr(x) \simeq nil **then** list('times, car(x), cadr(x))
 else list('times, car(x), times-tree(cdr(x))) **endif**

DEFINITION:

times-fringe(x)
= **if** listp(x) \wedge (car(x) = 'times)
 then append(times-fringe(cadr(x)), times-fringe(caddr(x)))
 else cons(x, nil) **endif**

DEFINITION:

```
or-zero-tree (x)
=  if x ≈ nil then '(false)
   elseif cdr (x) ≈ nil then list ('zerop, car (x))
   elseif caddr (x) ≈ nil
   then list ('or, list ('zerop, car (x)), list ('zerop, caddr (x)))
   else list ('or, list ('zerop, car (x)), or-zero-tree (cdr (x))) endif
```

DEFINITION:

```
and-not-zero-tree (x)
=  if x ≈ nil then '(true)
   elseif cdr (x) ≈ nil then list ('not, list ('zerop, car (x)))
   else list ('and,
             list ('not, list ('zerop, car (x))),
             and-not-zero-tree (cdr (x))) endif
```

THEOREM: numberp-eval\$-times

$(\text{car } (x) = \text{'times}) \rightarrow (\text{eval\$ } (\mathbf{t}, x, a) \in \mathbf{N})$

EVENT: Disable numberp-eval\$-times.

THEOREM: eval\$-times

$(\text{car } (x) = \text{'times})$
 $\rightarrow (\text{eval\$ } (\mathbf{t}, x, a) = (\text{eval\$ } (\mathbf{t}, \text{caddr } (x), a) * \text{eval\$ } (\mathbf{t}, \text{caddr } (x), a)))$

EVENT: Disable eval\$-times.

THEOREM: eval\$-or

$(\text{car } (x) = \text{'or})$
 $\rightarrow (\text{eval\$ } (\mathbf{t}, x, a) = (\text{eval\$ } (\mathbf{t}, \text{caddr } (x), a) \vee \text{eval\$ } (\mathbf{t}, \text{caddr } (x), a)))$

EVENT: Disable eval\$-or.

THEOREM: eval\$-equal

$(\text{car } (x) = \text{'equal})$
 $\rightarrow (\text{eval\$ } (\mathbf{t}, x, a) = (\text{eval\$ } (\mathbf{t}, \text{caddr } (x), a) = \text{eval\$ } (\mathbf{t}, \text{caddr } (x), a)))$

EVENT: Disable eval\$-equal.

THEOREM: eval\$-lessp

$(\text{car } (x) = \text{'lessp})$
 $\rightarrow (\text{eval\$ } (\mathbf{t}, x, a) = (\text{eval\$ } (\mathbf{t}, \text{caddr } (x), a) < \text{eval\$ } (\mathbf{t}, \text{caddr } (x), a)))$

EVENT: Disable eval\$-lessp.

THEOREM: eval\$-quotient

(car(x) = 'quotient)

→ (eval\$(t, x, a) = (eval\$(t, cadr(x), a) ÷ eval\$(t, caddr(x), a)))

EVENT: Disable eval\$-quotient.

THEOREM: eval\$-if

(car(x) = 'if)

→ (eval\$(t, x, a)

= if eval\$(t, cadr(x), a) then eval\$(t, caddr(x), a)
else eval\$(t, caddr(x), a) endif)

EVENT: Disable eval\$-if.

THEOREM: numberp-eval\$-times-tree

eval\$(t, times-tree(x), a) ∈ N

EVENT: Disable numberp-eval\$-times-tree.

THEOREM: lessp-times-arg1

(a ≠ 0) → (((a * x) < (a * y)) = (x < y))

THEOREM: infer-equality-from-not-lessp

((a ∈ N) ∧ (b ∈ N)) → (((a < b) ∧ (b < a)) = (a = b))

THEOREM: equal-times-arg1

(a ≠ 0) → (((a * x) = (a * y)) = (fix(x) = fix(y)))

EVENT: Disable equal-times-arg1.

THEOREM: equal-times-bridge

((a * b) = (c * (a * d))) = ((a ≈ 0) ∨ (fix(b) = (c * d)))

EVENT: Disable equal-times-bridge.

THEOREM: eval\$-times-member

(e ∈ x)

→ (eval\$(t, times-tree(x), a)

= (eval\$(t, e, a) * eval\$(t, times-tree(delete(e, x), a)))

EVENT: Disable eval\$-times-member.

THEOREM: zerop-makes-times-tree-zero
 $((\neg \text{eval}\$(\mathbf{t}, \text{and-not-zerop-tree}(x), a)) \wedge \text{subbagp}(x, y))$
 $\rightarrow (\text{eval}\$(\mathbf{t}, \text{times-tree}(y), a) = 0)$

EVENT: Disable zerop-makes-times-tree-zero.

THEOREM: or-zerop-tree-is-not-zerop-tree
 $\text{eval}\$(\mathbf{t}, \text{or-zerop-tree}(x), a) = (\neg \text{eval}\$(\mathbf{t}, \text{and-not-zerop-tree}(x), a))$

EVENT: Disable or-zerop-tree-is-not-zerop-tree.

THEOREM: zerop-makes-times-tree-zero2
 $(\text{eval}\$(\mathbf{t}, \text{or-zerop-tree}(x), a) \wedge \text{subbagp}(x, y))$
 $\rightarrow (\text{eval}\$(\mathbf{t}, \text{times-tree}(y), a) = 0)$

EVENT: Disable zerop-makes-times-tree-zero2.

THEOREM: times-tree-append
 $\text{eval}\$(\mathbf{t}, \text{times-tree}(\text{append}(x, y)), a)$
 $= (\text{eval}\$(\mathbf{t}, \text{times-tree}(x), a) * \text{eval}\$(\mathbf{t}, \text{times-tree}(y), a))$

EVENT: Disable times-tree-append.

THEOREM: times-tree-of-times-fringe
 $\text{eval}\$(\mathbf{t}, \text{times-tree}(\text{times-fringe}(x)), a) = \text{fix}(\text{eval}\$(\mathbf{t}, x, a))$

EVENT: Disable times-tree-of-times-fringe.

DEFINITION:

$\text{cancel-lessp-times}(x)$
 $=$ **if** $(\text{car}(x) = \text{'lessp})$
 $\quad \wedge (\text{caadr}(x) = \text{'times})$
 $\quad \wedge (\text{caaddr}(x) = \text{'times})$
then let $inboth$ **be** $\text{bagint}(\text{times-fringe}(\text{cadr}(x)),$
 $\quad \text{times-fringe}(\text{caddr}(x)))$
in
if $\text{listp}(inboth)$
then list $(\text{'and},$
 $\quad \text{and-not-zerop-tree}(inboth),$
 $\quad \text{list}(\text{'lessp},$
 $\quad \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe}(\text{cadr}(x)),$
 $\quad \quad \quad inboth)),$


```

times-tree (bagdiff (times-fringe (caddr (x)),
                    inboth))))
else x endif endlet
else x endif

```

THEOREM: eval\$-lessp-times-tree-bagdiff
 $(\text{subbagp}(x, y) \wedge \text{subbagp}(x, z) \wedge \text{eval}\$(\mathbf{t}, \text{and-not-zerop-tree}(x), a))$
 $\rightarrow ((\text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a)$
 $< \text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a))$
 $= (\text{eval}\$(\mathbf{t}, \text{times-tree}(y), a) < \text{eval}\$(\mathbf{t}, \text{times-tree}(z), a)))$

EVENT: Disable eval\$-lessp-times-tree-bagdiff.

THEOREM: zerop-makes-lessp-false-bridge
 $((\text{car}(x) = \text{'times})$
 $\wedge (\text{car}(y) = \text{'times})$
 $\wedge (\neg \text{eval}\$(\mathbf{t},$
 $\text{and-not-zerop-tree}(\text{bagint}(\text{times-fringe}(x), \text{times-fringe}(y)),$
 $a)))$
 $\rightarrow (((\text{eval}\$(\mathbf{t}, \text{cadr}(x), a) * \text{eval}\$(\mathbf{t}, \text{caddr}(x), a))$
 $< (\text{eval}\$(\mathbf{t}, \text{cadr}(y), a) * \text{eval}\$(\mathbf{t}, \text{caddr}(y), a)))$
 $= \mathbf{f})$

EVENT: Disable zerop-makes-lessp-false-bridge.

THEOREM: correctness-of-cancel-lessp-times
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-lessp-times}(x), a)$

DEFINITION:

```

cancel-equal-times (x)
= if (car (x) = 'equal)
    and (caadr (x) = 'times)
    and (caaddr (x) = 'times)
then let inboth be bagint (times-fringe (cadr (x)),
                           times-fringe (caddr (x)))
in
if listp (inboth)
then list ('or,
           or-zerop-tree (inboth),
           list ('equal,
                times-tree (bagdiff (times-fringe (cadr (x)),
                                     inboth)),
                times-tree (bagdiff (times-fringe (caddr (x)),
                                     inboth))))

```

else x endif endlet
else x endif

THEOREM: zerop-makes-equal-true-bridge

$$\begin{aligned} & ((\text{car } (x) = \text{'times}) \\ & \wedge (\text{car } (y) = \text{'times}) \\ & \wedge \text{eval}\$(\mathbf{t}, \text{or-zerop-tree}(\text{bagint}(\text{times-fringe } (x), \text{times-fringe } (y))), a)) \\ \rightarrow & (((\text{eval}\$(\mathbf{t}, \text{cadr } (x), a) * \text{eval}\$(\mathbf{t}, \text{caddr } (x), a)) \\ & = (\text{eval}\$(\mathbf{t}, \text{cadr } (y), a) * \text{eval}\$(\mathbf{t}, \text{caddr } (y), a))) \\ & = \mathbf{t}) \end{aligned}$$

EVENT: Disable zerop-makes-equal-true-bridge.

THEOREM: eval\$-equal-times-tree-bagdiff

$$\begin{aligned} & (\text{subbagp } (x, y) \wedge \text{subbagp } (x, z) \wedge (\neg \text{eval}\$(\mathbf{t}, \text{or-zerop-tree } (x), a))) \\ \rightarrow & ((\text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff } (y, x)), a) \\ & = \text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff } (z, x)), a)) \\ & = (\text{eval}\$(\mathbf{t}, \text{times-tree } (y), a) = \text{eval}\$(\mathbf{t}, \text{times-tree } (z), a))) \end{aligned}$$

EVENT: Disable eval\$-equal-times-tree-bagdiff.

THEOREM: cancel-equal-times-preserves-inequality

$$\begin{aligned} & (\text{subbagp } (z, x) \\ & \wedge \text{subbagp } (z, y) \\ & \wedge (\text{eval}\$(\mathbf{t}, \text{times-tree } (x), a) \neq \text{eval}\$(\mathbf{t}, \text{times-tree } (y), a))) \\ \rightarrow & (\text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff } (x, z)), a) \\ & \neq \text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff } (y, z)), a)) \end{aligned}$$

EVENT: Disable cancel-equal-times-preserves-inequality.

THEOREM: cancel-equal-times-preserves-inequality-bridge

$$\begin{aligned} & ((\text{car } (x) = \text{'times}) \\ & \wedge (\text{car } (y) = \text{'times}) \\ & \wedge (((\text{eval}\$(\mathbf{t}, \text{cadr } (x), a) * \text{eval}\$(\mathbf{t}, \text{caddr } (x), a)) \\ & \neq (\text{eval}\$(\mathbf{t}, \text{cadr } (y), a) * \text{eval}\$(\mathbf{t}, \text{caddr } (y), a)))) \\ \rightarrow & (\text{eval}\$(\mathbf{t}, \\ & \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe } (x), \\ & \quad \quad \text{bagint}(\text{times-fringe } (x), \text{times-fringe } (y))), \\ & \quad a) \\ & \neq \text{eval}\$(\mathbf{t}, \\ & \quad \text{times-tree}(\text{bagdiff}(\text{times-fringe } (y), \\ & \quad \quad \text{bagint}(\text{times-fringe } (x), \\ & \quad \quad \quad \text{times-fringe } (y))), \\ & \quad a)) \end{aligned}$$

EVENT: Disable cancel-equal-times-preserves-inequality-bridge.

THEOREM: correctness-of-cancel-equal-times

$\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-equal-times}(x), a)$

```
; Define the available theory of multiplication. To get the list of
; events to put in the theory, evaluate the following form in NQTHM at
; this point in the script. This form lists all lemmas which are
; globally enabled, and which have non-null lemma type.
;
; (remove-if-not (function (lambda (x)
;   (and (member x (lemmas))
;   (not (assoc x disabled-lemmas))
;   (not (null (nth 2 (get x 'event))))))
; (not (member x (nth 2 (get 'addition 'event))))))
;   chronology)
```

EVENT: Let us define the theory *multiplication* to consist of the following events: equal-times-0, equal-times-1, equal-sub1-0, times-zero, times-add1, commutativity-of-times, times-distributes-over-plus, commutativity2-of-times, associativity-of-times, times-distributes-over-difference, times-quotient, times-1-arg1, lessp-times1, lessp-times2, lessp-times3, lessp-plus-times1, lessp-plus-times2, lessp-1-times, correctness-of-cancel-lessp-times, correctness-of-cancel-equal-times.

```
; ----- REMAINDER -----
```

THEOREM: lessp-remainder

$((x \mathbf{mod} y) < y) = (y \neq 0)$

THEOREM: remainder-noop

$(a < b) \rightarrow ((a \mathbf{mod} b) = \text{fix}(a))$

THEOREM: remainder-of-non-number

$(a \notin \mathbf{N}) \rightarrow ((a \mathbf{mod} n) = (0 \mathbf{mod} n))$

THEOREM: remainder-zero

$(x \simeq 0) \rightarrow ((y \mathbf{mod} x) = \text{fix}(y))$

THEOREM: plus-remainder-times-quotient

$((x \mathbf{mod} y) + (y * (x \div y))) = \text{fix}(x)$

EVENT: Disable plus-remainder-times-quotient.

THEOREM: remainder-quotient-elim
 $((y \neq 0) \wedge (x \in \mathbf{N})) \rightarrow (((x \bmod y) + (y * (x \div y))) = x)$

```

; (lemma remainder-sub1 (rewrite)
;   (implies (and (not (zerop a))
;     (not (zerop b)))
; (equal (remainder (sub1 a) b)
;   (if (equal (remainder a b) 0)
;     (sub1 b)
;     (sub1 (remainder a b)))))
;   ((enable lessp-remainder
; remainder-noop
; remainder-quotient-elim)
; (enable-theory addition)
; (induct (remainder a b))))

```

THEOREM: remainder-add1
 $((a \bmod b) = 0) \rightarrow ((1 + a) \bmod b) = (1 \bmod b)$

THEOREM: remainder-plus-proof
 $((b \bmod c) = 0) \rightarrow ((a + b) \bmod c) = (a \bmod c)$

THEOREM: remainder-plus
 $((a \bmod c) = 0)$
 $\rightarrow (((a + b) \bmod c) = (b \bmod c))$
 $\wedge (((b + a) \bmod c) = (b \bmod c))$
 $\wedge (((x + y + a) \bmod c) = ((x + y) \bmod c))$

THEOREM: equal-remainder-plus-0-proof
 $((a \bmod c) = 0) \rightarrow (((a + b) \bmod c) = 0) = ((b \bmod c) = 0)$

THEOREM: equal-remainder-plus-0
 $((a \bmod c) = 0)$
 $\rightarrow (((((a + b) \bmod c) = 0) = ((b \bmod c) = 0))$
 $\wedge (((((b + a) \bmod c) = 0) = ((b \bmod c) = 0))$
 $\wedge (((((x + y + a) \bmod c) = 0)$
 $= (((x + y) \bmod c) = 0)))$

THEOREM: equal-remainder-plus-remainder-proof
 $(a < c) \rightarrow (((a + b) \bmod c) = (b \bmod c)) = (a \simeq 0)$

THEOREM: equal-remainder-plus-remainder
 $(a < c)$
 $\rightarrow (((((a + b) \bmod c) = (b \bmod c)) = (a \simeq 0))$
 $\wedge (((((b + a) \bmod c) = (b \bmod c)) = (a \simeq 0)))$

EVENT: Disable equal-remainder-plus-remainder.

THEOREM: remainder-times1-proof

$$((b \bmod c) = 0) \rightarrow (((a * b) \bmod c) = 0)$$

THEOREM: remainder-times1

$$\begin{aligned} & ((b \bmod c) = 0) \\ \rightarrow & (((a * b) \bmod c) = 0) \wedge (((b * a) \bmod c) = 0) \end{aligned}$$

THEOREM: remainder-times1-instance-proof

$$((x * y) \bmod y) = 0$$

THEOREM: remainder-times1-instance

$$(((x * y) \bmod y) = 0) \wedge (((x * y) \bmod x) = 0)$$

THEOREM: remainder-times-times-proof

$$((x * y) \bmod (x * z)) = (x * (y \bmod z))$$

THEOREM: remainder-times-times

$$\begin{aligned} & (((x * y) \bmod (x * z)) = (x * (y \bmod z))) \\ \wedge & (((x * z) \bmod (y * z)) = ((x \bmod y) * z)) \end{aligned}$$

EVENT: Disable remainder-times-times.

THEOREM: remainder-times2-proof

$$((a \bmod z) = 0) \rightarrow ((a \bmod (z * y)) = (z * ((a \div z) \bmod y)))$$

THEOREM: remainder-times2

$$\begin{aligned} & ((a \bmod z) = 0) \\ \rightarrow & (((a \bmod (y * z)) = (z * ((a \div z) \bmod y))) \\ & \wedge ((a \bmod (z * y)) = (z * ((a \div z) \bmod y)))) \end{aligned}$$

THEOREM: remainder-times2-instance

$$\begin{aligned} & (((x * y) \bmod (x * z)) = (x * (y \bmod z))) \\ \wedge & (((x * z) \bmod (y * z)) = ((x \bmod y) * z)) \end{aligned}$$

THEOREM: remainder-difference1

$$\begin{aligned} & ((a \bmod c) = (b \bmod c)) \\ \rightarrow & (((a - b) \bmod c) = ((a \bmod c) - (b \bmod c))) \end{aligned}$$

DEFINITION:

double-remainder-induction (a, b, c)

```
= if  $c \simeq 0$  then 0
   elseif  $a < c$  then 0
   elseif  $b < c$  then 0
   else double-remainder-induction( $a - c, b - c, c$ ) endif
```

THEOREM: remainder-difference2
 $((a \bmod c) = 0) \wedge ((b \bmod c) \neq 0)$
 $\rightarrow ((a - b) \bmod c)$
 $= \text{if } b < a \text{ then } c - (b \bmod c)$
 $\text{else } 0 \text{ endif}$

THEOREM: remainder-difference3
 $((b \bmod c) = 0) \wedge ((a \bmod c) \neq 0)$
 $\rightarrow ((a - b) \bmod c)$
 $= \text{if } b < a \text{ then } a \bmod c$
 $\text{else } 0 \text{ endif}$

EVENT: Disable remainder-difference3.

THEOREM: equal-remainder-difference-0
 $((a - b) \bmod c) = 0$
 $= \text{if } b \leq a \text{ then } (a \bmod c) = (b \bmod c)$
 $\text{else } \mathbf{t} \text{ endif}$

EVENT: Disable equal-remainder-difference-0.

THEOREM: lessp-plus-fact
 $((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (b < c) \wedge (a < x)$
 $\rightarrow ((a + b) < c) = \mathbf{t}$

EVENT: Disable lessp-plus-fact.

THEOREM: remainder-plus-fact
 $((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (a < x)$
 $\rightarrow (((a + b) \bmod c) = (a + (b \bmod c)))$

THEOREM: remainder-plus-times-times-proof
 $(a < b)$
 $\rightarrow (((a + (b * c)) \bmod (b * d))$
 $= (a + ((b * c) \bmod (b * d))))$

THEOREM: remainder-plus-times-times
 $(a < b)$
 $\rightarrow (((((a + (b * c)) \bmod (b * d))$
 $= (a + ((b * c) \bmod (b * d))))$
 $\wedge (((a + (c * b)) \bmod (d * b))$
 $= (a + ((c * b) \bmod (d * b))))))$

```
; REMAINDER-PLUS-TIMES-TIMES-INSTANCE is the completion of the rules
; TIMES-DISTRIBUTES-OVER-PLUS, REMAINDER-TIMES-TIMES and REMAINDER-PLUS-TIMES-TIMES
```

THEOREM: remainder-plus-times-times-instance

$$\begin{aligned}
& (a < b) \\
\rightarrow & \left(\left(\left((a + (b * c) + (b * d)) \bmod (b * e) \right) \right. \right. \\
& \quad = \left. \left(a + (b * ((c + d) \bmod e)) \right) \right) \\
& \quad \wedge \left(\left((a + (c * b) + (d * b)) \bmod (e * b) \right) \right. \\
& \quad \quad = \left. \left(a + (b * ((c + d) \bmod e)) \right) \right)
\end{aligned}$$

THEOREM: remainder-remainder

$$((b \bmod a) = 0) \rightarrow (((n \bmod b) \bmod a) = (n \bmod a))$$

THEOREM: remainder-1-arg1

$$\begin{aligned}
& (1 \bmod x) \\
= & \text{ if } x = 1 \text{ then } 0 \\
& \text{ else } 1 \text{ endif}
\end{aligned}$$

THEOREM: remainder-1-arg2

$$(y \bmod 1) = 0$$

THEOREM: remainder-x-x

$$(x \bmod x) = 0$$

THEOREM: transitivity-of-divides

$$(((a \bmod b) = 0) \wedge ((b \bmod c) = 0)) \rightarrow ((a \bmod c) = 0)$$

```
; Define the available theory of remainder. To get the list of
; events to put in the theory, evaluate the following form in NQTHM at
; this point in the script. This form lists all lemmas which are
; globally enabled, and which have non-null lemma type.
```

```
;
;
; (let ((lemmas (lemmas)))
;   (remove-if-not (function (lambda (x)
;     (and (member x lemmas)
;         (not (assoc x disabled-lemmas))
;         (not (null (nth 2 (get x 'event))))
;         (not (member x (nth 2 (get 'addition 'event))))
;         (not (member x (nth 2 (get 'multiplication 'event)))))))))
;   chronology))
```

EVENT: Let us define the theory *remainders* to consist of the following events:
 lessp-remainder, remainder-noop, remainder-of-non-number, remainder-zero, remainder-
 quotient-elim, remainder-add1, remainder-plus, equal-remainder-plus-0, remainder-
 times1, remainder-times1-instance, remainder-times2, remainder-times2-instance,
 remainder-difference1, remainder-difference2, remainder-plus-times-times, remainder-
 plus-times-times-instance, remainder-remainder, remainder-1-arg1, remainder-
 1-arg2, remainder-x-x.

; ----- QUOTIENT, DIVIDES -----

THEOREM: quotient-noop
 $(b = 1) \rightarrow ((a \div b) = \text{fix}(a))$

THEOREM: quotient-of-non-number
 $(a \notin \mathbf{N}) \rightarrow ((a \div n) = (0 \div n))$

THEOREM: quotient-zero
 $(x \simeq 0) \rightarrow ((y \div x) = 0)$

THEOREM: quotient-add1
 $((a \bmod b) = 0)$
 $\rightarrow (((1 + a) \div b)$
 $\quad = \text{if } b = 1 \text{ then } 1 + (a \div b)$
 $\quad \text{else } a \div b \text{ endif})$

THEOREM: equal-quotient-0
 $((a \div b) = 0) = ((b \simeq 0) \vee (a < b))$

THEOREM: quotient-sub1
 $((a \not\simeq 0) \wedge (b \not\simeq 0))$
 $\rightarrow (((a - 1) \div b)$
 $\quad = \text{if } (a \bmod b) = 0 \text{ then } (a \div b) - 1$
 $\quad \text{else } a \div b \text{ endif})$

THEOREM: quotient-plus-proof
 $((b \bmod c) = 0) \rightarrow (((a + b) \div c) = ((a \div c) + (b \div c)))$

THEOREM: quotient-plus
 $((a \bmod c) = 0)$
 $\rightarrow (((((a + b) \div c) = ((a \div c) + (b \div c)))$
 $\quad \wedge (((b + a) \div c) = ((a \div c) + (b \div c)))$
 $\quad \wedge (((x + y + a) \div c)$
 $\quad \quad = (((x + y) \div c) + (a \div c))))$

; I need QUOTIENT-TIMES-INSTANCE to prove the more general QUOTIENT-TIMES,
; but I want QUOTIENT-TIMES-INSTANCE to be tried first (i.e. come after
; QUOTIENT-TIMES in the event list.) So first, prove QUOTIENT-TIMES-INSTANCE-TEMP,
; then prove QUOTIENT-TIMES, and finally give QUOTIENT-TIMES-INSTANCE.

THEOREM: quotient-times-instance-temp-proof

$$\begin{aligned} & ((y * x) \div y) \\ = & \text{ if } y \simeq 0 \text{ then } 0 \\ & \text{ else fix } (x) \text{ endif} \end{aligned}$$

THEOREM: quotient-times-instance-temp

$$\begin{aligned} & (((y * x) \div y) \\ = & \text{ if } y \simeq 0 \text{ then } 0 \\ & \text{ else fix } (x) \text{ endif}) \\ \wedge & (((x * y) \div y) \\ = & \text{ if } y \simeq 0 \text{ then } 0 \\ & \text{ else fix } (x) \text{ endif}) \end{aligned}$$

EVENT: Disable quotient-times-instance-temp.

THEOREM: quotient-times-proof

$$((a \text{ mod } c) = 0) \rightarrow (((a * b) \div c) = (b * (a \div c)))$$

THEOREM: quotient-times

$$\begin{aligned} & ((a \text{ mod } c) = 0) \\ \rightarrow & (((a * b) \div c) = (b * (a \div c))) \\ & \wedge (((b * a) \div c) = (b * (a \div c))) \end{aligned}$$

THEOREM: quotient-times-instance

$$\begin{aligned} & (((y * x) \div y) \\ = & \text{ if } y \simeq 0 \text{ then } 0 \\ & \text{ else fix } (x) \text{ endif}) \\ \wedge & (((x * y) \div y) \\ = & \text{ if } y \simeq 0 \text{ then } 0 \\ & \text{ else fix } (x) \text{ endif}) \end{aligned}$$

THEOREM: quotient-times-times-proof

$$\begin{aligned} & ((x * y) \div (x * z)) \\ = & \text{ if } x \simeq 0 \text{ then } 0 \\ & \text{ else } y \div z \text{ endif} \end{aligned}$$

THEOREM: quotient-times-times

$$\begin{aligned} & (((x * y) \div (x * z)) \\ = & \text{ if } x \simeq 0 \text{ then } 0 \end{aligned}$$

$$\begin{aligned}
& \text{else } y \div z \text{ endif}) \\
\wedge & \ ((x * z) \div (y * z)) \\
& = \text{if } z \simeq 0 \text{ then } 0 \\
& \quad \text{else } x \div y \text{ endif})
\end{aligned}$$

EVENT: Disable quotient-times-times.

THEOREM: quotient-difference1

$$\begin{aligned}
& ((a \bmod c) = (b \bmod c)) \\
\rightarrow & \ (((a - b) \div c) = ((a \div c) - (b \div c)))
\end{aligned}$$

THEOREM: quotient-lessp-arg1

$$(a < b) \rightarrow ((a \div b) = 0)$$

THEOREM: quotient-difference2

$$\begin{aligned}
& (((a \bmod c) = 0) \wedge ((b \bmod c) \neq 0)) \\
\rightarrow & \ (((a - b) \div c) \\
& \quad = \text{if } b < a \text{ then } (a \div c) - (1 + (b \div c)) \\
& \quad \text{else } 0 \text{ endif})
\end{aligned}$$

THEOREM: quotient-difference3

$$\begin{aligned}
& (((b \bmod c) = 0) \wedge ((a \bmod c) \neq 0)) \\
\rightarrow & \ (((a - b) \div c) \\
& \quad = \text{if } b < a \text{ then } (a \div c) - (b \div c) \\
& \quad \text{else } 0 \text{ endif})
\end{aligned}$$

THEOREM: remainder-equals-its-first-argument

$$(a = (a \bmod b)) = ((a \in \mathbf{N}) \wedge ((b \simeq 0) \vee (a < b)))$$

EVENT: Disable remainder-equals-its-first-argument.

THEOREM: quotient-remainder-times

$$((x \bmod (a * b)) \div a) = ((x \div a) \bmod b)$$

THEOREM: quotient-remainder

$$((c \bmod a) = 0) \rightarrow (((b \bmod c) \div a) = ((b \div a) \bmod (c \div a)))$$

THEOREM: quotient-remainder-instance

$$((x \bmod (a * b)) \div a) = ((x \div a) \bmod b)$$

THEOREM: quotient-plus-fact

$$\begin{aligned}
& (((b \bmod x) = 0) \wedge ((c \bmod x) = 0) \wedge (a < x)) \\
\rightarrow & \ (((a + b) \div c) = (b \div c))
\end{aligned}$$

THEOREM: quotient-plus-times-times-proof

$$(a < b) \\ \rightarrow (((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d)))$$

THEOREM: quotient-plus-times-times

$$(a < b) \\ \rightarrow (((((a + (b * c)) \div (b * d)) = ((b * c) \div (b * d))) \\ \wedge (((a + (b * c)) \div (b * d)) \\ = ((b * c) \div (b * d))))))$$

; QUOTIENT-PLUS-TIMES-TIMES-INSTANCE is the completion of the rules
; QUOTIENT-TIMES-TIMES, QUOTIENT-PLUS-TIMES-TIMES and TIMES-DISTRIBUTES-OVER-PLUS

THEOREM: quotient-plus-times-times-instance

$$(a < b) \\ \rightarrow (((((a + (b * c)) + (b * d)) \div (b * e)) \\ = \text{if } b \simeq 0 \text{ then } 0 \\ \text{else } (c + d) \div e \text{ endif}) \\ \wedge (((a + (c * b)) + (d * b)) \div (e * b)) \\ = \text{if } b \simeq 0 \text{ then } 0 \\ \text{else } (d + c) \div e \text{ endif}))$$

THEOREM: quotient-quotient

$$((b \div a) \div c) = (b \div (a * c))$$

THEOREM: leq-quotient

$$(a < b) \rightarrow ((a \div c) \leq (b \div c))$$

THEOREM: quotient-1-arg2

$$(n \div 1) = \text{fix}(n)$$

THEOREM: quotient-1-arg1-casesplit

$$(n \simeq 0) \vee (n = 1) \vee (1 < n)$$

THEOREM: quotient-1-arg1

$$(1 \div n) \\ = \text{if } n = 1 \text{ then } 1 \\ \text{else } 0 \text{ endif}$$

THEOREM: quotient-x-x

$$(x \neq 0) \rightarrow ((x \div x) = 1)$$

THEOREM: lessp-quotient

$$((i \div j) < i) = ((i \neq 0) \wedge (j \neq 1))$$

```

;; Metalemma to cancel quotient-times expressions

;; ex.
;; (quotient (times a b) (times c (times d a))) ->
;;                                     (if (not (zerop a))
;;                                         (quotient (fix b) (times c d))
;;                                         (zero))
;;
;;

```

DEFINITION:

```

cancel-quotient-times (x)
=  if (car (x) = 'quotient)
    ^ (caadr (x) = 'times)
    ^ (caaddr (x) = 'times)
    then let inboth be bagint (times-fringe (cadr (x)),
                              times-fringe (caddr (x)))
          in
          if listp (inboth)
          then list ('if,
                    and-not-zerop-tree (inboth),
                    list ('quotient,
                          times-tree (bagdiff (times-fringe (cadr (x)),
                                                inboth)),
                          times-tree (bagdiff (times-fringe (caddr (x)),
                                                inboth))),
                    '(zero))
          else x endif endlet
    else x endif

```

THEOREM: zerop-makes-quotient-zero-bridge

```

((car (x) = 'times)
 ^ (car (y) = 'times)
 ^ (¬ eval$ (t,
             and-not-zerop-tree (bagint (times-fringe (x), times-fringe (y)),
                                     a)))
→ (((eval$ (t, cadr (x), a) * eval$ (t, caddr (x), a))
    ÷ (eval$ (t, cadr (y), a) * eval$ (t, caddr (y), a)))
   = 0)

```

EVENT: Disable zerop-makes-quotient-zero-bridge.

THEOREM: eval\$-quotient-times-tree-bagdiff

```

(subbagp (x, y) ^ subbagp (x, z) ^ eval$ (t, and-not-zerop-tree (x), a))

```

$$\begin{aligned} \rightarrow & ((\text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff}(y, x)), a) \\ & \div \text{eval}\$(\mathbf{t}, \text{times-tree}(\text{bagdiff}(z, x)), a)) \\ & = (\text{eval}\$(\mathbf{t}, \text{times-tree}(y), a) \div \text{eval}\$(\mathbf{t}, \text{times-tree}(z), a))) \end{aligned}$$

EVENT: Disable eval\$-quotient-times-tree-bagdiff.

THEOREM: correctness-of-cancel-quotient-times
 $\text{eval}\$(\mathbf{t}, x, a) = \text{eval}\$(\mathbf{t}, \text{cancel-quotient-times}(x), a)$

```

; Define the available theory of quotient. To get the list of events to
; put in the theory, evaluate the following form in NQTHM at this point
; in the script. This form lists all lemmas which are globally enabled,
; and which have non-null lemma type.
;
;
; (let ((lemmas (lemmas)))
;   (remove-if-not (function (lambda (x)
;     (and (member x lemmas)
;         (not (assoc x disabled-lemmas))
;         (not (null (nth 2 (get x 'event))))
;         (not (member x (nth 2 (get 'addition 'event))))
;         (not (member x (nth 2 (get 'multiplication 'event))))
;         (not (member x (nth 2 (get 'remainders 'event))))))
;     chronology))

```

EVENT: Let us define the theory *quotients* to consist of the following events: quotient-noop, quotient-of-non-number, quotient-zero, quotient-add1, equal-quotient-0, quotient-sub1, quotient-plus, quotient-times, quotient-times-instance, quotient-difference1, quotient-lessp-arg1, quotient-difference2, quotient-difference3, quotient-remainder-times, quotient-remainder, quotient-remainder-instance, quotient-plus-times-times, quotient-plus-times-times-instance, quotient-quotient, quotient-1-arg2, quotient-1-arg1, quotient-x-x, lessp-quotient, correctness-of-cancel-quotient-times.

```
;;; exp, log, and gcd
```

DEFINITION:
 $\text{exp}(i, j)$
 $=$ **if** $j \simeq 0$ **then** 1
 else $i * \text{exp}(i, j - 1)$ **endif**

DEFINITION:

$\log(\text{base}, n)$
= **if** $\text{base} < 2$ **then** 0
 elseif $n \simeq 0$ **then** 0
 else $1 + \log(\text{base}, n \div \text{base})$ **endif**

DEFINITION:

$\text{gcd}(x, y)$
= **if** $x \simeq 0$ **then** $\text{fix}(y)$
 elseif $y \simeq 0$ **then** x
 elseif $x < y$ **then** $\text{gcd}(x, y - x)$
 else $\text{gcd}(x - y, y)$ **endif**

THEOREM: remainder-exp

$(k \not\simeq 0) \rightarrow ((\text{exp}(n, k) \bmod n) = 0)$

DEFINITION:

double-number-induction(i, j)
= **if** $i \simeq 0$ **then** 0
 elseif $j \simeq 0$ **then** 0
 else double-number-induction($i - 1, j - 1$) **endif**

THEOREM: remainder-exp-exp

$(i \leq j) \rightarrow ((\text{exp}(a, j) \bmod \text{exp}(a, i)) = 0)$

THEOREM: quotient-exp

$(k \not\simeq 0)$
 $\rightarrow ((\text{exp}(n, k) \div n)$
 = **if** $n \simeq 0$ **then** 0
 else $\text{exp}(n, k - 1)$ **endif)**

THEOREM: exp-zero

$(k \simeq 0) \rightarrow (\text{exp}(n, k) = 1)$

THEOREM: exp-add1

$\text{exp}(n, 1 + k) = (n * \text{exp}(n, k))$

THEOREM: exp-plus

$\text{exp}(i, j + k) = (\text{exp}(i, j) * \text{exp}(i, k))$

THEOREM: exp-0-arg1

$\text{exp}(0, k)$
= **if** $k \simeq 0$ **then** 1
 else 0 **endif**

THEOREM: exp-1-arg1

$\text{exp}(1, k) = 1$

THEOREM: exp-0-arg2

$$\exp(n, 0) = 1$$

THEOREM: exp-times

$$\exp(i * j, k) = (\exp(i, k) * \exp(j, k))$$

THEOREM: exp-exp

$$\exp(\exp(i, j), k) = \exp(i, j * k)$$

THEOREM: equal-exp-0

$$(\exp(n, k) = 0) = ((n \simeq 0) \wedge (k \not\simeq 0))$$

THEOREM: equal-exp-1

$$\begin{aligned} &(\exp(n, k) = 1) \\ &= \text{if } k \simeq 0 \text{ then } \mathbf{t} \\ &\quad \text{else } n = 1 \text{ endif} \end{aligned}$$

THEOREM: exp-difference

$$((c \leq b) \wedge (a \not\simeq 0)) \rightarrow (\exp(a, b - c) = (\exp(a, b) \div \exp(a, c)))$$

EVENT: Let us define the theory *exponentiation* to consist of the following events: equal-exp-0, equal-exp-1, exp-exp, exp-add1, exp-times, exp-1-arg1, exp-zero, exp-0-arg2, exp-0-arg1, exp-difference, exp-plus, quotient-exp, remainder-exp-exp, remainder-exp.

THEOREM: equal-log-0

$$(\log(\text{base}, n) = 0) = ((\text{base} < 2) \vee (n \simeq 0))$$

THEOREM: log-0

$$(n \simeq 0) \rightarrow (\log(\text{base}, n) = 0)$$

THEOREM: log-1

$$(1 < \text{base}) \rightarrow (\log(\text{base}, 1) = 1)$$

DEFINITION:

double-log-induction(*base*, *a*, *b*)

$$\begin{aligned} &= \text{if } \text{base} < 2 \text{ then } 0 \\ &\quad \text{elseif } a \simeq 0 \text{ then } 0 \\ &\quad \text{elseif } b \simeq 0 \text{ then } 0 \\ &\quad \text{else double-log-induction}(\text{base}, a \div \text{base}, b \div \text{base}) \text{ endif} \end{aligned}$$

THEOREM: leq-log-log

$$(n \leq m) \rightarrow (\log(c, n) \leq \log(c, m))$$

THEOREM: log-quotient

$$(1 < c) \rightarrow (\log(c, n \div c) = (\log(c, n) - 1))$$

THEOREM: log-quotient-times-proof

$$(1 < c) \rightarrow (\log(c, n \div (c * m)) = (\log(c, n \div m) - 1))$$

THEOREM: log-quotient-times

$$(1 < c) \\ \rightarrow ((\log(c, n \div (c * m)) = (\log(c, n \div m) - 1)) \\ \wedge (\log(c, n \div (m * c)) = (\log(c, n \div m) - 1)))$$

THEOREM: log-quotient-exp

$$(1 < c) \rightarrow (\log(c, n \div \exp(c, m)) = (\log(c, n) - m))$$

THEOREM: log-times-proof

$$((1 < c) \wedge (n \neq 0)) \rightarrow (\log(c, c * n) = (1 + \log(c, n)))$$

THEOREM: log-times

$$((1 < c) \wedge (n \neq 0)) \\ \rightarrow ((\log(c, c * n) = (1 + \log(c, n))) \\ \wedge (\log(c, n * c) = (1 + \log(c, n))))$$

THEOREM: log-times-exp-proof

$$((1 < c) \wedge (n \neq 0)) \rightarrow (\log(c, n * \exp(c, m)) = (\log(c, n) + m))$$

THEOREM: log-times-exp

$$((1 < c) \wedge (n \neq 0)) \\ \rightarrow ((\log(c, n * \exp(c, m)) = (\log(c, n) + m)) \\ \wedge (\log(c, \exp(c, m) * n) = (\log(c, n) + m)))$$

THEOREM: log-exp

$$(1 < c) \rightarrow (\log(c, \exp(c, n)) = (1 + n))$$

EVENT: Let us define the theory *logs* to consist of the following events: log-exp, log-times-exp, log-times, log-quotient-exp, log-quotient-times, log-quotient, log-1, log-0, equal-log-0, exp-exp.

THEOREM: commutativity-of-gcd

$$\text{gcd}(b, a) = \text{gcd}(a, b)$$

DEFINITION:

single-number-induction (n)

= **if** $n \simeq 0$ **then** 0
 else single-number-induction ($n - 1$) **endif**

THEOREM: gcd-0

$$(\text{gcd}(0, x) = \text{fix}(x)) \wedge (\text{gcd}(x, 0) = \text{fix}(x))$$

THEOREM: gcd-1

$$(\text{gcd}(1, x) = 1) \wedge (\text{gcd}(x, 1) = 1)$$

THEOREM: equal-gcd-0

$$(\text{gcd}(a, b) = 0) = ((a \simeq 0) \wedge (b \simeq 0))$$

THEOREM: lessp-gcd

$$(b \not\leq 0) \rightarrow (((b < \text{gcd}(a, b)) = \mathbf{f}) \wedge ((b < \text{gcd}(b, a)) = \mathbf{f}))$$

THEOREM: gcd-plus-instance-temp-proof

$$\text{gcd}(a, a + b) = \text{gcd}(a, b)$$

THEOREM: gcd-plus-instance-temp

$$(\text{gcd}(a, a + b) = \text{gcd}(a, b)) \wedge (\text{gcd}(a, b + a) = \text{gcd}(a, b))$$

THEOREM: gcd-plus-proof

$$((b \mathbf{mod} a) = 0) \rightarrow (\text{gcd}(a, b + c) = \text{gcd}(a, c))$$

THEOREM: gcd-plus

$$\begin{aligned} & ((b \mathbf{mod} a) = 0) \\ \rightarrow & ((\text{gcd}(a, b + c) = \text{gcd}(a, c)) \\ & \wedge (\text{gcd}(a, c + b) = \text{gcd}(a, c)) \\ & \wedge (\text{gcd}(b + c, a) = \text{gcd}(a, c)) \\ & \wedge (\text{gcd}(c + b, a) = \text{gcd}(a, c))) \end{aligned}$$

THEOREM: gcd-plus-instance

$$(\text{gcd}(a, a + b) = \text{gcd}(a, b)) \wedge (\text{gcd}(a, b + a) = \text{gcd}(a, b))$$

THEOREM: remainder-gcd

$$((a \mathbf{mod} \text{gcd}(a, b)) = 0) \wedge ((b \mathbf{mod} \text{gcd}(a, b)) = 0)$$

THEOREM: distributivity-of-times-over-gcd-proof

$$\text{gcd}(x * z, y * z) = (z * \text{gcd}(x, y))$$

THEOREM: distributivity-of-times-over-gcd

$$\begin{aligned} & (\text{gcd}(x * z, y * z) = (z * \text{gcd}(x, y))) \\ \wedge & (\text{gcd}(z * x, y * z) = (z * \text{gcd}(x, y))) \\ \wedge & (\text{gcd}(x * z, z * y) = (z * \text{gcd}(x, y))) \\ \wedge & (\text{gcd}(z * x, z * y) = (z * \text{gcd}(x, y))) \end{aligned}$$

THEOREM: gcd-is-the-greatest

$$\begin{aligned} & ((x \not\leq 0) \wedge (y \not\leq 0) \wedge ((x \mathbf{mod} z) = 0) \wedge ((y \mathbf{mod} z) = 0)) \\ \rightarrow & (z \leq \text{gcd}(x, y)) \end{aligned}$$

THEOREM: common-divisor-divides-gcd

$$(((x \mathbf{mod} z) = 0) \wedge ((y \mathbf{mod} z) = 0)) \rightarrow ((\text{gcd}(x, y) \mathbf{mod} z) = 0)$$

; We prove ASSOCIATIVITY-OF-GCD and COMMUTATIVITY2-OF-GCD roughly the same way.
; Use GCD-IS-THE-GREATEST twice to show that each side of the equality is
; less than or equal to the other side.

THEOREM: associativity-of-gcd-zero-case
 $((a \simeq 0) \vee (b \simeq 0) \vee (c \simeq 0)) \rightarrow (\text{gcd}(\text{gcd}(a, b), c) = \text{gcd}(a, \text{gcd}(b, c)))$

THEOREM: associativity-of-gcd
 $\text{gcd}(\text{gcd}(a, b), c) = \text{gcd}(a, \text{gcd}(b, c))$

THEOREM: commutativity2-of-gcd-zero-case
 $((a \simeq 0) \vee (b \simeq 0) \vee (c \simeq 0)) \rightarrow (\text{gcd}(b, \text{gcd}(a, c)) = \text{gcd}(a, \text{gcd}(b, c)))$

THEOREM: commutativity2-of-gcd
 $\text{gcd}(b, \text{gcd}(a, c)) = \text{gcd}(a, \text{gcd}(b, c))$

THEOREM: gcd-x-x
 $\text{gcd}(x, x) = \text{fix}(x)$

THEOREM: gcd-idempotence
 $(\text{gcd}(x, \text{gcd}(x, y)) = \text{gcd}(x, y)) \wedge (\text{gcd}(y, \text{gcd}(x, y)) = \text{gcd}(x, y))$

EVENT: Let us define the theory *gcds* to consist of the following events: commutativity2-of-gcd, associativity-of-gcd, common-divisor-divides-gcd, distributivity-of-times-over-gcd, lessp-gcd, equal-gcd-0, gcd-0, gcd-idempotence, gcd-x-x, remainder-gcd, gcd-plus, gcd-plus-instance, gcd-1, commutativity-of-gcd.

EVENT: Let us define the theory *naturals* to consist of the following events: addition, multiplication, remainders, quotients, exponentiation, logs, gcds.

EVENT: Make the library "**naturals**" and compile it.

Index

- addition, 10
- and-not-zero-tree, 14, 16, 17, 28
- associativity-of-gcd, 34
- associativity-of-gcd-zero-case, 34
- associativity-of-plus, 2
- associativity-of-times, 11

- bagdiff, 5, 7–10, 16–18, 28, 29
- bagint, 7–10, 16–18, 28
- bridge-to-subbagp-implies-plus-t
ree-greatereqp, 5

- cadr-eval\$list, 6
- cancel-difference-plus, 8
- cancel-equal-plus, 6, 7
- cancel-equal-times, 17, 19
- cancel-equal-times-preserves-ineq
uality, 18
 - uality-bridge, 18
- cancel-lessp-plus, 9, 10
- cancel-lessp-times, 16, 17
- cancel-quotient-times, 28, 29
- common-divisor-divides-gcd, 33
- commutativity-of-gcd, 32
- commutativity-of-plus, 2
- commutativity-of-times, 11
- commutativity2-of-gcd, 34
- commutativity2-of-gcd-zero-case, 34
- commutativity2-of-plus, 2
- commutativity2-of-times, 11
- correctness-of-cancel-difference
-plus, 8
- correctness-of-cancel-equal-plu
s, 7
- correctness-of-cancel-equal-time
s, 19
- correctness-of-cancel-lessp-plu
s, 10
- correctness-of-cancel-lessp-time
s, 17
- correctness-of-cancel-quotient-ti
mes, 29

- delete, 5, 7, 8, 10, 15
- diff-diff-arg1, 3
- diff-diff-arg2, 3
- diff-diff-diff, 4
- diff-sub1-arg2, 3
- difference-add1-arg2, 8
- difference-cancellation, 2
- difference-difference-arg1, 9
- difference-difference-arg2, 9
- difference-elim, 8
- difference-leq-arg1, 8
- difference-lessp-arg1, 4
- difference-plus-cancellation, 3
- difference-plus-cancellation-pr
oof, 3
- difference-plus-plus-cancellati
on, 3
 - on-hack, 3
 - on-proof, 3
- difference-sub1-arg2, 9
- difference-x-x, 9
- distributivity-of-times-over-gc
d, 33
 - d-proof, 33
- double-log-induction, 31
- double-number-induction, 30
- double-remainder-induction, 21

- equal-difference-0, 2
- equal-exp-0, 31
- equal-exp-1, 31
- equal-gcd-0, 33
- equal-log-0, 31
- equal-plus-0, 1
- equal-quotient-0, 24
- equal-remainder-difference-0, 22
- equal-remainder-plus-0, 20
- equal-remainder-plus-0-proof, 20
- equal-remainder-plus-remainder, 20

- equal-remainder-plus-remainder-p
 - roof, 20
- equal-sub1-0, 11
- equal-times-0, 10
- equal-times-1, 11
- equal-times-arg1, 15
- equal-times-bridge, 15
- eval\$-equal, 14
- eval\$-equal-times-tree-bagdiff, 18
- eval\$-if, 15
- eval\$-lessp, 14
- eval\$-lessp-times-tree-bagdiff, 17
- eval\$-or, 14
- eval\$-plus-tree-append, 5
- eval\$-quote, 6
- eval\$-quotient, 15
- eval\$-quotient-times-tree-bagdi
 - ff, 28
- eval\$-times, 14
- eval\$-times-member, 15
- exp, 29–32
 - exp-0-arg1, 30
 - exp-0-arg2, 31
 - exp-1-arg1, 30
 - exp-add1, 30
 - exp-difference, 31
 - exp-exp, 31
 - exp-plus, 30
 - exp-times, 31
 - exp-zero, 30
- exponentiation, 31

- gcd, 30, 32–34
 - gcd-0, 32
 - gcd-1, 33
 - gcd-idempotence, 34
 - gcd-is-the-greatest, 33
 - gcd-plus, 33
 - gcd-plus-instance, 33
 - gcd-plus-instance-temp, 33
 - gcd-plus-instance-temp-proof, 33
 - gcd-plus-proof, 33
 - gcd-x-x, 34
 - gcds, 34

- infer-equality-from-not-lessp, 15

- leq-log-log, 31
- leq-quotient, 27
- lessp-1-times, 13
- lessp-difference-cancellation, 9
- lessp-gcd, 33
- lessp-plus-fact, 22
- lessp-plus-times-proof, 12
- lessp-plus-times1, 13
- lessp-plus-times2, 13
- lessp-quotient, 27
- lessp-remainder, 19
- lessp-times-arg1, 15
- lessp-times-cancellation-proof, 12
- lessp-times-cancellation1, 12
- lessp-times1, 12
- lessp-times1-proof, 12
- lessp-times2, 12
- lessp-times2-proof, 12
- lessp-times3, 12
- lessp-times3-proof1, 12
- lessp-times3-proof2, 12
- listp-eval\$, 6
- log, 30–32
 - log-0, 31
 - log-1, 31
 - log-exp, 32
 - log-quotient, 31
 - log-quotient-exp, 32
 - log-quotient-times, 32
 - log-quotient-times-proof, 32
 - log-times, 32
 - log-times-exp, 32
 - log-times-exp-proof, 32
 - log-times-proof, 32
- logs, 32

- member-implies-numberp, 6
- member-implies-plus-tree-greate
 - reqp, 5
- multiplication, 19

- naturals, 34

- numberp-eval\$-bridge, 5
- numberp-eval\$-plus, 4
- numberp-eval\$-plus-tree, 4
- numberp-eval\$-times, 14
- numberp-eval\$-times-tree, 15

- or-zero-tree, 14, 16–18
- or-zero-tree-is-not-zero-tree, 16

- plus-add1-arg1, 2
- plus-add1-arg2, 2
- plus-cancellation, 1
- plus-difference-arg1, 2
- plus-difference-arg2, 2
- plus-fringe, 4–10
- plus-remainder-times-quotient, 19
- plus-tree, 4–10
- plus-tree-bagdiff, 5
- plus-tree-delete, 5
- plus-tree-plus-fringe, 6
- plus-zero-arg2, 2

- quotient-1-arg1, 27
- quotient-1-arg1-casesplit, 27
- quotient-1-arg2, 27
- quotient-add1, 24
- quotient-difference1, 26
- quotient-difference2, 26
- quotient-difference3, 26
- quotient-exp, 30
- quotient-lessp-arg1, 26
- quotient-noop, 24
- quotient-of-non-number, 24
- quotient-plus, 24
- quotient-plus-fact, 26
- quotient-plus-proof, 24
- quotient-plus-times-times, 27
- quotient-plus-times-times-insta-
nce, 27
- quotient-plus-times-times-proof, 27
- quotient-quotient, 27
- quotient-remainder, 26
- quotient-remainder-instance, 26
- quotient-remainder-times, 26

- quotient-sub1, 24
- quotient-times, 25
- quotient-times-instance, 25
- quotient-times-instance-temp, 25
- quotient-times-instance-temp-pr
oof, 25
- quotient-times-proof, 25
- quotient-times-times, 25
- quotient-times-times-proof, 25
- quotient-x-x, 27
- quotient-zero, 24
- quotients, 29

- remainder-1-arg1, 23
- remainder-1-arg2, 23
- remainder-add1, 20
- remainder-difference1, 21
- remainder-difference2, 22
- remainder-difference3, 22
- remainder-equals-its-first-argu-
ment, 26
- remainder-exp, 30
- remainder-exp-exp, 30
- remainder-gcd, 33
- remainder-noop, 19
- remainder-of-non-number, 19
- remainder-plus, 20
- remainder-plus-fact, 22
- remainder-plus-proof, 20
- remainder-plus-times-times, 22
- remainder-plus-times-times-inst
ance, 23
- remainder-plus-times-times-proo
f, 22
- remainder-quotient-elim, 20
- remainder-remainder, 23
- remainder-times-times, 21
- remainder-times-times-proof, 21
- remainder-times1, 21
- remainder-times1-instance, 21
- remainder-times1-instance-proof, 21
- remainder-times1-proof, 21
- remainder-times2, 21
- remainder-times2-instance, 21

remainder-times2-proof, 21
remainder-x-x, 23
remainder-zero, 19
remainders, 24

single-number-induction, 32
subbagp, 5, 16–18, 28
subbagp-implies-plus-tree-greate
 reqp, 5

times-1-arg1, 12
times-add1, 11
times-distributes-over-differen
 ce, 11
 ce-proof, 11
times-distributes-over-plus, 11
times-distributes-over-plus-pro
 of, 11
times-fringe, 13, 16–18, 28
times-quotient, 12
times-quotient-proof, 11
times-tree, 13, 15–18, 28, 29
times-tree-append, 16
times-tree-of-times-fringe, 16
times-zero, 11
transitivity-of-divides, 23

zerop-makes-equal-true-bridge, 18
zerop-makes-lessp-false-bridge, 17
zerop-makes-quotient-zero-bridge, 28
zerop-makes-times-tree-zero, 16
zerop-makes-times-tree-zero2, 16